

# **Отчет по лабораторной работе №7**

**Дисциплина: архитектура компьютера**

Мазурский Александр Дмитриевич

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>7</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>8</b>
4.1	Реализация переходов в NASM . . . . .	8
4.2	Изучение структуры файла листинга . . . . .	13
4.3	Задания для самостоятельной работы . . . . .	16
<b>5</b>	<b>Выводы</b>	<b>22</b>
	<b>Список литературы</b>	<b>23</b>

## Список иллюстраций

4.1	Создание каталога и файла для программы . . . . .	8
4.2	Сохранение программы . . . . .	9
4.3	Запуск программы . . . . .	9
4.4	Изменение программы . . . . .	10
4.5	Запуск измененной программы . . . . .	11
4.6	Изменение программы . . . . .	12
4.7	Проверка изменений . . . . .	12
4.8	Сохранение новой программы . . . . .	13
4.9	Проверка программы из листинга . . . . .	13
4.10	Проверка файла листинга . . . . .	14
4.11	Удаление операнда из программы . . . . .	15
4.12	Просмотр ошибки в файле листинга . . . . .	16
4.13	Первая программа самостоятельной работы . . . . .	17
4.14	Проверка работы первой программы . . . . .	19
4.15	Вторая программа самостоятельной работы . . . . .	20

## **Список таблиц**

# 1 Цель работы

Изучение команд условного и безусловного переходов. Приобретение навыков написания программ с использованием переходов. Знакомство с назначением и структурой файла листинга.

## 2 Задание

1. Реализация переходов в NASM
2. Изучение структуры файлов листинга
3. Самостоятельное написание программ по материалам лабораторной работы

### **3 Теоретическое введение**

Для реализации ветвлений в ассемблере используются так называемые команды передачи управления или команды перехода. Можно выделить 2 типа переходов: • условный переход – выполнение или не выполнение перехода в определенную точку программы в зависимости от проверки условия. • безусловный переход – выполнение передачи управления в определенную точку программы без каких-либо условий.

## 4 Выполнение лабораторной работы

### 4.1 Реализация переходов в NASM

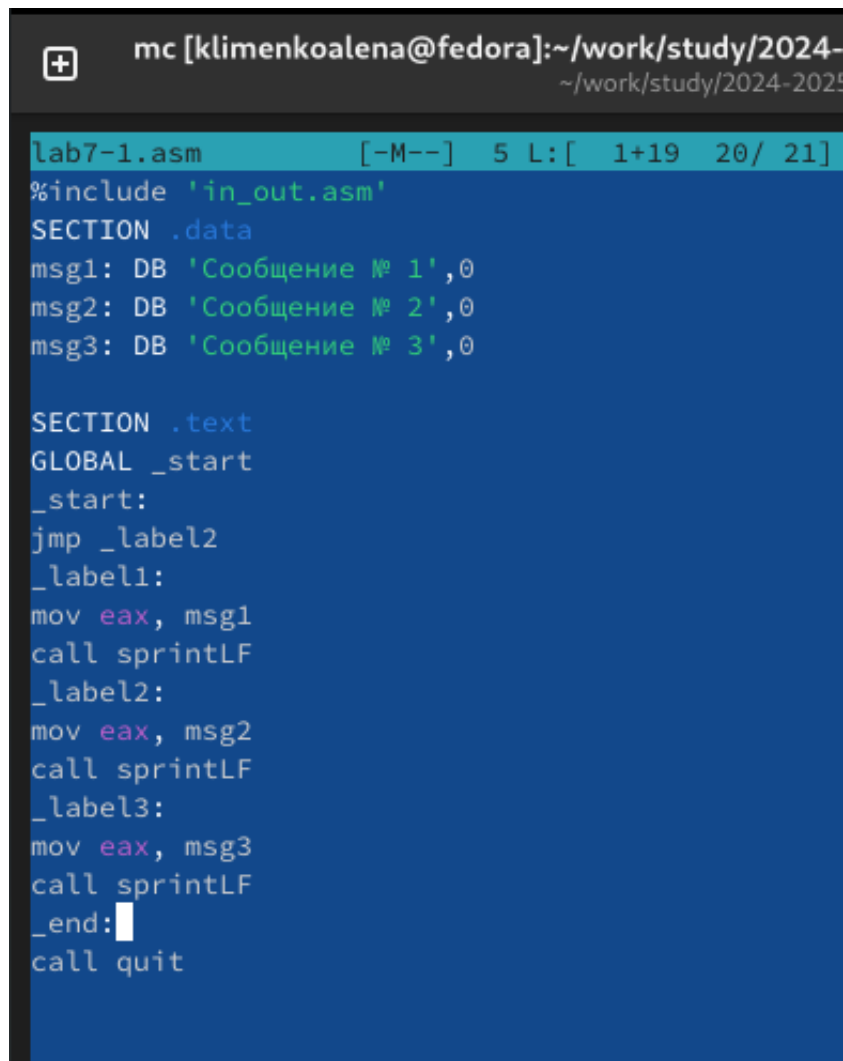
Создаю каталог для программ лабораторной работы №7 (рис. 4.1).

```
klimenkoalenafedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04/report$ mkdir ~/work/study/2024-2025/Архитектура\ компьютера/arch-pc/lab07
klimenkoalenafedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04/report$ cd ~/work/arch-pc/lab07
bash: cd: /home/klimenkoalena/work/arch-pc/lab07: Нет такого файла или каталога
klimenkoalenafedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04/report$ cd ~/work/study/2024-2025/Архитектура\ компьютера/arch-pc/lab07
klimenkoalenafedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ touch lab7-1.asm
klimenkoalenafedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ls
lab7-1.asm
klimenkoalenafedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$
```

Рис. 4.1: Создание каталога и файла для программы

Копирую код из листинга в файл будущей программы. (рис. 4.2).



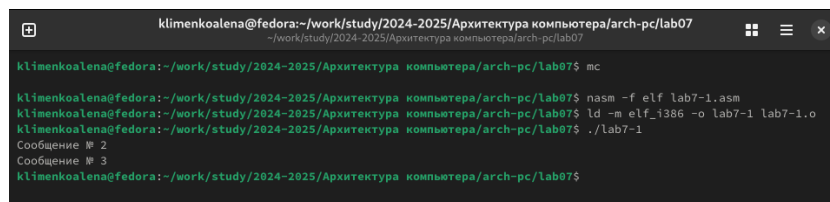


```
mc [klimenkoalena@fedora]:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07
lab7-1.asm [-M--] 5 L:[ 1+19 20/ 21]
%include 'in_out.asm'
SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0

SECTION .text
GLOBAL _start
_start:
jmp _label2
_label1:
mov eax, msg1
call sprintLF
_label2:
mov eax, msg2
call sprintLF
_label3:
mov eax, msg3
call sprintLF
_end:
call quit
```

Рис. 4.2: Сохранение программы

При запуске программы я убедился в том, что безусловный переход действительно изменяет порядок выполнения инструкций (рис. 4.3).



```
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ mc
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ nasm -f elf lab7-1.asm
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$
```

Рис. 4.3: Запуск программы

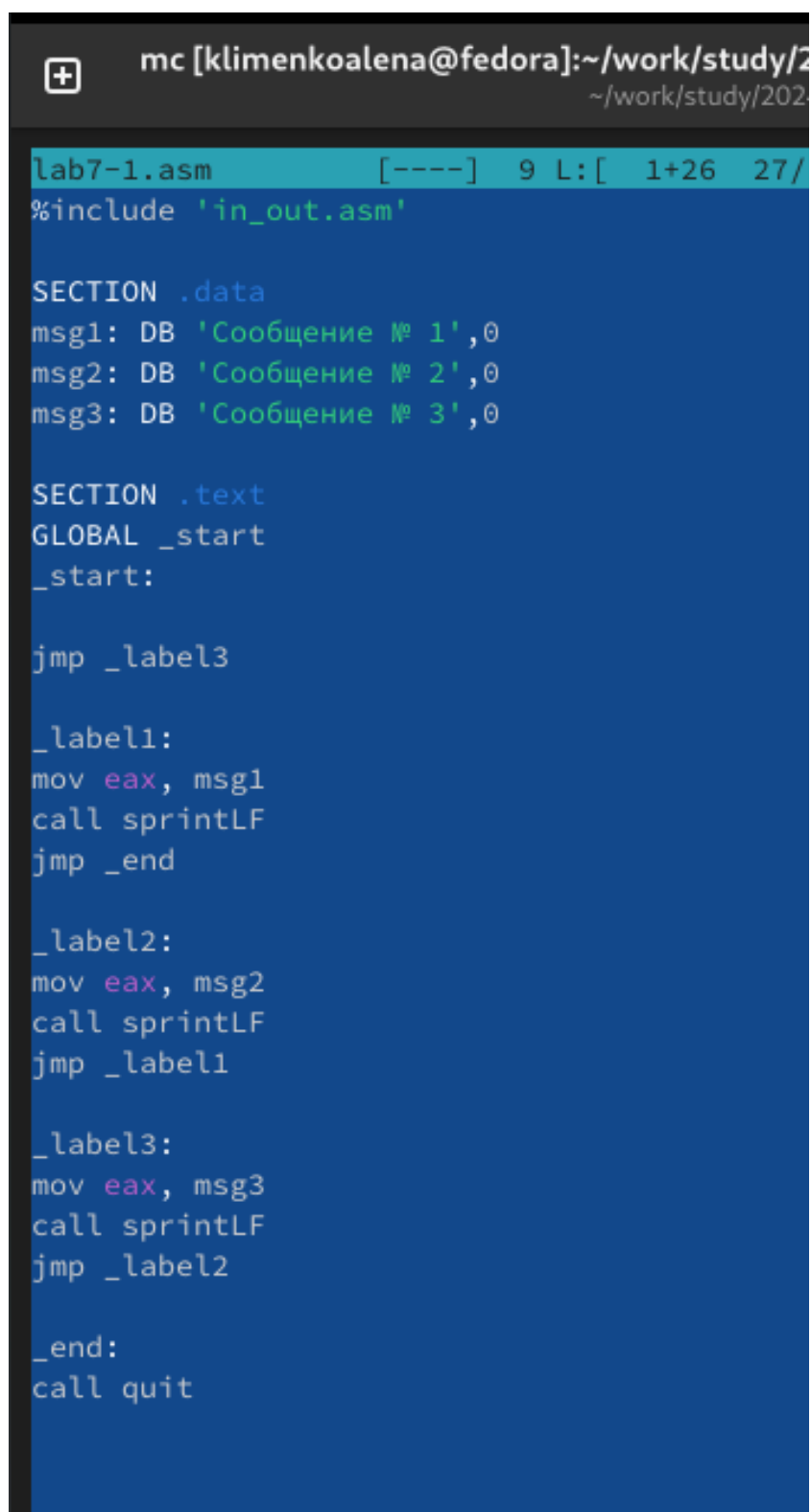
Изменяю программу таким образом, чтобы поменялся порядок выполнения

функций (рис. 4.4).

```
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ nasm -f elf lab7-1.asm
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 1
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$
```

Рис. 4.4: Изменение программы

Запускаю программу и проверяю, что примененные изменения верны (рис. 4.5).



```
mc [klimenkoalena@fedora]:~/work/study/2
~/work/study/202
lab7-1.asm [----] 9 L: [ 1+26 27/
%include 'in_out.asm'

SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0

SECTION .text
GLOBAL _start
_start:

jmp _label3

_label1:
mov eax, msg1
call sprintf
jmp _end

_label2:
mov eax, msg2
call sprintf
jmp _label1

_label3:
mov eax, msg3
call sprintf
jmp _label2

_end:
call quit
```

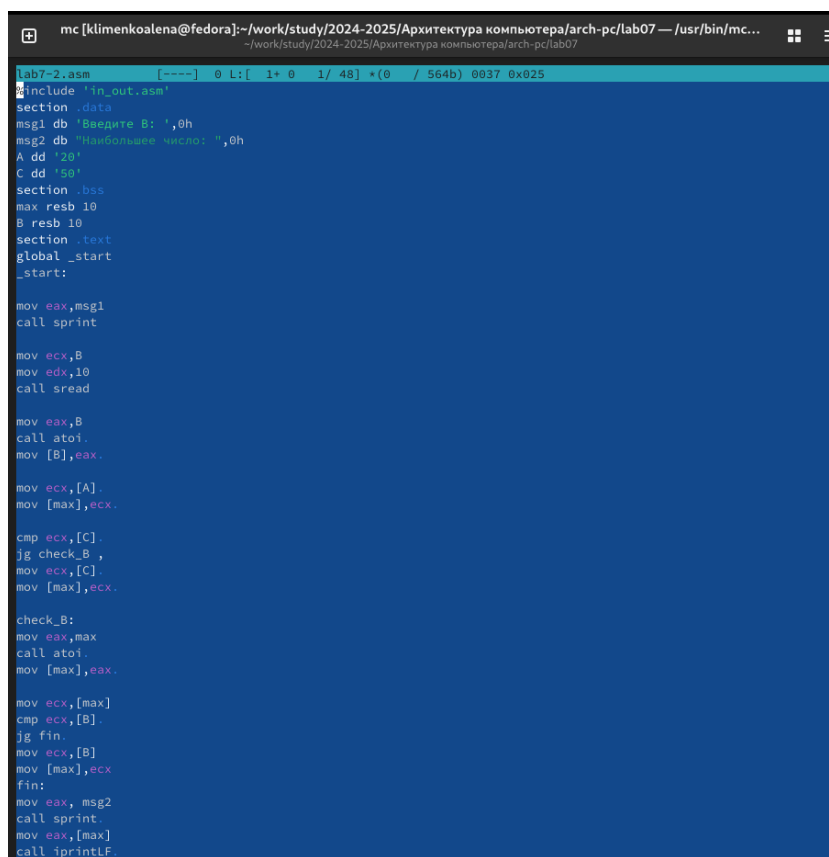
Рис. 4.5: Запуск измененной программы

Теперь изменяю текст программы так, чтобы все три сообщения вывелись в обратном порядке (рис. 4.6).

```
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ nasm -f elf lab7-1.asm
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
```

Рис. 4.6: Изменение программы

Работа выполнена корректно, программа в нужном мне порядке выводит сообщения (рис. 4.7).



```
lab7-2.asm  [----]  0 L: [ 1* 0 1/ 48] *(0 / 564b) 0037 0x025
#include 'in_out.asm'
section .data
msg1 db 'Введите B: ',0h
msg2 db "Наибольшее число: ",0h
A dd '20'
C dd '50'
section .bss
max resb 10
B resb 10
section .text
global _start
_start:

mov eax,msg1
call sprint

mov ecx,B
mov edx,10
call sread

mov eax,B
call atoi
mov [B],eax

mov ecx,[A]
mov [max],ecx

cmp ecx,[C]
jg check_B
mov ecx,[C]
mov [max],ecx

check_B:
mov eax,max
call atoi
mov [max],eax

mov ecx,[max]
cmp ecx,[B]
jg fin
mov ecx,[B]
mov [max],ecx
fin:
mov eax,msg2
call sprint
mov eax,[max]
call iprintLF
```

Рис. 4.7: Проверка изменений

Создаю новый рабочий файл и вставляю в него код из следующего листинга (рис. 4.8).

```

klimenkoalenafedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ nasm -f elf lab7-2.asm
klimenkoalenafedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
klimenkoalenafedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ./lab7-2
Введите B: 5
Наибольшее число: 50
klimenkoalenafedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ./lab7-2
Введите B: 51
Наибольшее число: 51
klimenkoalenafedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ./lab7-2
Введите B: 50
Наибольшее число: 50
klimenkoalenafedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$

```

Рис. 4.8: Сохранение новой программы

Программа выводит значение переменной с максимальным значением, проверяя работу программы с разными входными данными (рис. 4.9).

```

~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07/lab7-2.lst - Mousepad
Файл  Правка  Поиск  Просмотр  Документ  Помощь
1      %include 'in_out.asm'
2      <1> ;----- slen -----
3      <1> ; Функция вычисления длины сообщения
4      <1> slen:
5      00000000 53      <1> push    ebx
6      00000001 89C3    <1> mov     ebx, eax
7      <1>
8      <1> nextchar:
9      00000003 803800    <1> cmp     byte [eax], 0
10     00000006 7403      <1> jz      finished
11     00000008 40         <1> inc     eax
12     00000009 EBF8      <1> jmp     nextchar
13     <1>
14     <1> finished:
15     0000000B 29D8      <1> sub     eax, ebx
16     0000000D 5B         <1> pop     ebx
17     0000000E C3         <1> ret
18     <1>
19     <1> ;----- sprint -----
20     <1> ; Функция печати сообщения
21     <1> ; входные данные: mov eax,<message>
22     <1> sprint:
23     0000000F 52         <1> push    edx
24     00000010 51         <1> push    ecx
25     00000011 53         <1> push    ebx
26     00000012 50         <1> push    eax
27     00000013 E8E8FFFF    <1> call    slen
28     <1>
29     00000018 89C2      <1> mov     edx, eax
30     0000001A 58         <1> pop     eax
31     <1>
32     0000001B 89C1      <1> mov     ecx, eax
33     0000001D B801000000 <1> mov     ebx, 1

```

Рис. 4.9: Проверка программы из листинга

## 4.2 Изучение структуры файла листинга

Создаю файл листинга с помощью флага -l команды nasm и открываю его с помощью текстового редактора mousepad (рис. 4.10).

```

mov eax,msg1
call sprint

mov ecx,B
mov edx,10
call sread

mov eax,B
call atoi
mov [B],eax

mov ecx,[A]
mov [max],ecx

cmp ecx,[C]
jg check_B ,
mov ecx,[C]
mov [max],ecx

```

Рис. 4.10: Проверка файла листинга

Первое значение в файле листинга - номер строки, и он может вовсе не совпадать с номером строки изначального файла. Второе вхождение - адрес, смещение машинного кода относительно начала текущего сегмента, затем непосредственно идет сам машинный код, а заключает строку исходный текст программы с комментариями.

Удаляю один операнд из случайной инструкции, чтобы проверить поведение файла листинга в дальнейшем (рис. 4.11).

```

20 00000017 00000000 mov ebx,0
19 000000FC E842FFFFFF call sread
20
21 00000101 B8[0A000000] mov eax,B
22 00000106 E891FFFFFF call atoi
23 mov [B],eax
23 ***** error: invalid combination of opcode and operands
24
25 0000010B 8B0D[35000000] mov ecx,[A]
26 00000111 890D[00000000] mov [max],ecx
27
28 00000117 3B0D[30000000] cmp ecx,[C]

```

Рис. 4.11: Удаление операнда из программы

В новом файле листинга показывает ошибку, которая возникла при попытке трансляции файла. Никакие выходные файлы при этом помимо файла листинга не создаются. (рис. 4.12).





4.13).

```
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ./lab7-3
Наименьшее число: 59
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$
```

Рис. 4.13: Первая программа самостоятельной работы

Код первой программы:

```
%include 'in_out.asm'

SECTION .data
msg1 db 'Введите B: ', 0h
msg2 db 'Наименьшее число: ', 0h
A dd '24'
C dd '15'

SECTION .bss
min resb 10
B resb 10

SECTION .text
GLOBAL _start
_start:

mov eax, msg1
call sprint

mov ecx, B
mov edx, 10
call sread
```

```

mov eax, B
call atoi
mov [B], eax

mov ecx, [A]
mov [min], ecx

cmp ecx, [C]
jg check_B
mov ecx, [C]
mov [min], ecx

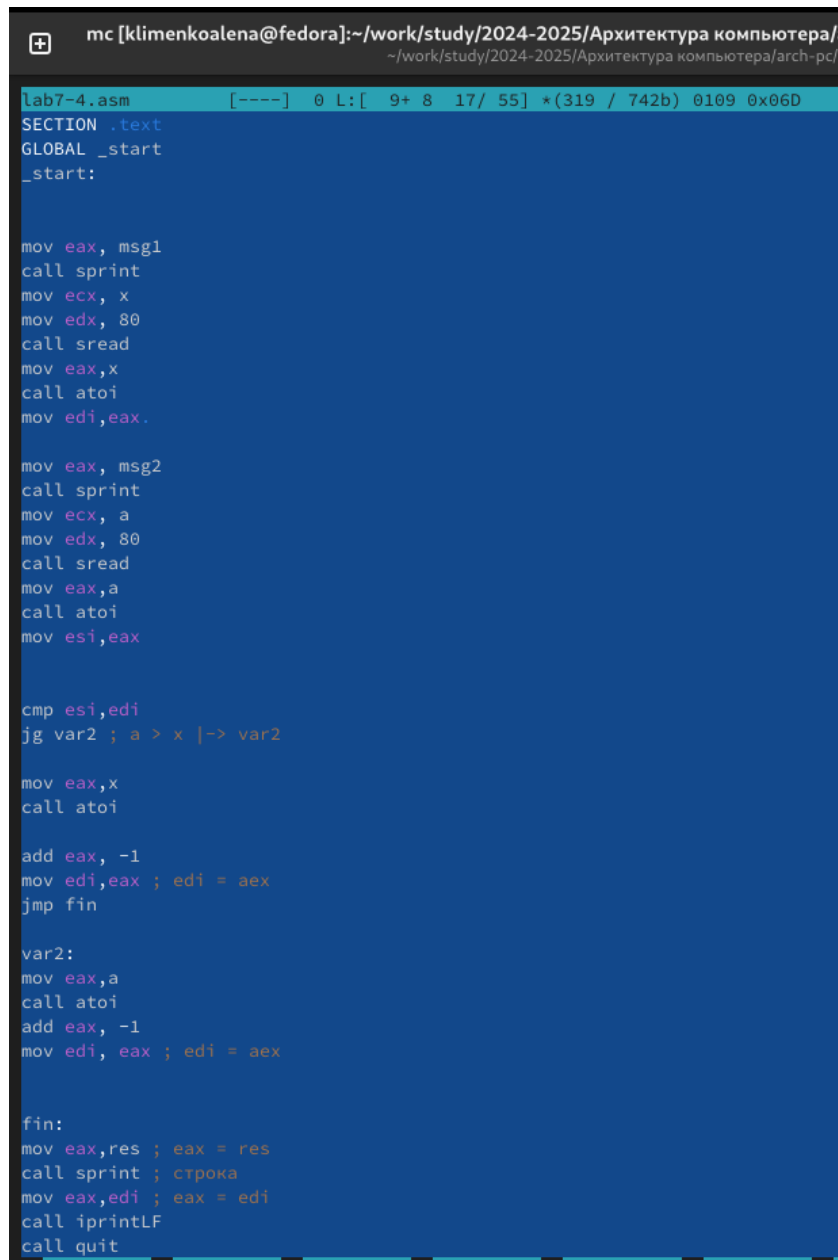
check_B:
mov eax, min
call atoi
mov [min], eax

mov ecx, [min]
cmp ecx, [B]
jb fin
mov ecx, [B]
mov [min], ecx

fin:
mov eax, msg2
call sprint
mov eax, [min]
call iprintLF
call quit

```

Проверяю корректность написания первой программы (рис. 4.14).



```
mc [klimenkoalena@fedora]:~/work/study/2024-2025/Архитектура компьютера/...
lab7-4.asm [----] 0 L: [ 9+ 8 17/ 55] *(319 / 742b) 0109 0x06D
SECTION .text
GLOBAL _start
_start:

mov eax, msg1
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi
mov edi, eax.

mov eax, msg2
call sprint
mov ecx, a
mov edx, 80
call sread
mov eax, a
call atoi
mov esi, eax

cmp esi, edi
jg var2 ; a > x |-> var2

mov eax, x
call atoi

add eax, -1
mov edi, eax ; edi = aex
jmp fin

var2:
mov eax, a
call atoi
add eax, -1
mov edi, eax ; edi = aex

fin:
mov eax, res ; eax = res
call sprint ; строка
mov eax, edi ; eax = edi
call iprintLF
call quit
```

Рис. 4.14: Проверка работы первой программы

Пишу программу, которая будет вычислять значение заданной функции согласно моему варианту для введенных с клавиатуры переменных а и х (рис. 4.15).

```
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ nasm -f elf lab7-4.asm
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ld -m elf_i386 -o lab7-4 lab7-4.o
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ./lab7-4
Введите значение переменной x: 5
Введите значение переменной a: 7
Результат: 6
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ./lab7-4
Введите значение переменной x: 6
Введите значение переменной a: 4
Результат: 5
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$
```

Рис. 4.15: Вторая программа самостоятельной работы

Код второй программы:

```
%include 'in_out.asm'

SECTION .data
msg_x: DB 'Введите значение переменной x: ', 0
msg_a: DB 'Введите значение переменной a: ', 0
res: DB 'Результат: ', 0

SECTION .bss
x: RESB 80
a: RESB 80

SECTION .text
GLOBAL _start
_start:
mov eax, msg_x
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi
mov edi, eax

mov eax, msg_a
call sprint
mov ecx, a
```

```

mov edx, 80
call sread
mov eax, a
call atoi
mov esi, eax

cmp edi, esi
jle add_values
mov eax, esi
jmp print_result

add_values:
mov eax, edi
add eax, esi

print_result:
mov edi, eax
mov eax, res
call sprint
mov eax, edi
call iprintLF
call quit

```

## **5 Выводы**

При выполнении лабораторной работы я изучил команды условных и безусловных переходов, а также приобрел навыки написания программ с использованием переходов, познакомился с назначением и структурой файлов листинга.

# Список литературы

1. Курс на ТУИС
2. Лабораторная работа №7
3. Программирование на языке ассемблера NASM Столяров А. В.